Desarrollo de Aplicaciones Telemáticas, examen práctico, 26 de junio de 2026
Grado en Ingeniería en Tecnologías de la Telecomunicación
Grado en Ingeniería en Sistemas de la Telecomunicación
Universidad Rey Juan Carlos

Instrucciones

- Emplea lo visto en clase y solo lo visto en clase (que puedes consultar en las transparencias). No será válido ni JavaScript más antiguo ni más avanzado. Usa el modo estricto. Si algún ejercicio tiene errores de sintaxis y/o no hace nada, su nota será nula.

Ejercicio 1. (4.0 puntos)

Escribe un programa en JavaScript llamado *horarios.js* según la siguiente especificación:

1. Tendrá una función llamada *crear_horarios*, que podrá llamar a otras funciones y que recibirá como parámetro una lista de sublistas.
2. El primer elemento de cada sublista será un nombre de asignatura, y los siguientes elementos, nombres de los alumnos matriculados [1]. Ejemplo:

```
let matricula =
[
    [ "DAT", "Ana", "Juan", "Pedro" ],
    [ "AST", "Ana", "Maria" ],
    [ "IO", "Pedro", "Maria" ],
    [ "Proyectos", "Juan", "Maria" ],
    [ "Satelites", "Ana", "Pedro" ],
    [ "Radiacion", "Juan" ]
];
```

3. La función *crear_horarios* deberá construir y devolver un *plain_object*. Cada clave de este objeto será el nombre de un alumno. El valor asociado será una lista con los nombres de las asignaturas en las que está matriculado dicho alumno.

   Para el ejemplo anterior, el resultado sería:

```
{
    "Ana": [ "DAT", "AST", "Satelites" ],
    "Juan": [ "DAT", "Proyectos", "Radiacion" ],
    "Pedro": [ "DAT", "IO", "Satelites" ],
    "Maria": [ "AST", "IO", "Proyectos" ]
}
```

4. El orden de las asignaturas en cada lista del objeto de salida, deberá coincidir con el orden en el que aparecen en la lista de entrada.
5. La función deberá comprobar que:

   a) Recibe exactamente un parámetro.

   b) El parámetro recibido es una lista.

   c) Cada elemento de la lista es una sublista.

   Si alguna de las comprobaciones anteriores falla, la función deberá lanzar un objeto excepción.

[1] Este es un mal diseño, muy simplificado. Lo adecuado sería usar identificadores de asignatura y de alumno.

6. En un programa real habría que comprobar que siempre hay una asignatura, que la asignatura no es una cadena vacía, que los alumnos no sean cadenas vacías, etc. Pero puedes ignorar todo esto. También puedes ignorar cómo responde tu programa si sucede alguno de estos errores.

7. El programa usará exactamente la lista *matricula* del ejemplo, llamará a *crea_horarios* y mostrará el resultado en pantalla. Aunque, naturalmente, tendrá que funcionar para cualquier lista que cumpla con el formato.

8. Añade también ejemplos de llamadas con listas erróneas. Pero que sean comentarios, de forma que el profesor pueda probarlo borrando las barras de comentario.

Ejercicio 2. (6.0 puntos)

Modifica tu aplicación de auto-pedido según la siguiente especificación:

1. Aparecerá una tabla adicional, llamada *eventos*.

2. En la tabla de eventos se añadirá una fila cada vez que el usuario pida un producto o cada vez que el usuario cancele un producto de su pedido.

3. Cada una de estas filas tendrá tres columnas:

   a) Fecha y hora en que ocurre el evento.
   No importa idioma ni formato, pero que sea legible.

   b) Nombre de producto. Cerveza, hamburguesa, etc.
   Para obtener la máxima puntuación, deberán actualizarse tanto los eventos pasados como los futuros cuando cambie el idioma. No obstante, si los eventos pasados permanecen en el idioma que tenían cuando fueron creados, la penalización no será importante.

   c) Descripción del evento.
   Descripción de qué sucede y qué consecuencias tiene. Ejemplo:

   - `+1 -> 4`
     (Esto significa que el cliente pide 1 producto más, eran 3, ahora son 4).
   - `-1 -> 0`
     (Esto significa que el cliente pide 1 producto menos, era 1, ahora son 0).

4. La aplicación tendrá dos modos de funcionamiento: *usuario* y *administrador*. La tabla de eventos solo se verá en modo administrador.

5. Un botón permitrá cambiar de modo. En modo *usuario*, el texto del botón será *administrador*. Y viceversa. Esto es, el botón indica a qué modo se pasará al pulsarse, no el estado actual.